Tools To Assist With HLA Pedagogy

David Stratton; Dr Phil Smith

University of Ballarat, SITMS
d.stratton@ballarat.edu.au
p.smith@ballarat.edu.au

Dr John Wharington

Defence Science and Technology Organisation
john.wharington@dsto.defence.gov.au

Abstract. The High Level Architecture (HLA) offers a dramatic extension of reuse for distributed simulation components. At the same time the HLA represents a significant training and education challenge if adoption of the architecture is to proceed at an adequate pace. In this context it is appropriate to consider tools and techniques that support effective pedagogy for HLA. This paper describes two innovations that have proved useful in short courses for HLA developers. The first supports scripting of basic HLA interactions so that a significant first encounter with HLA can proceed without the cognitive overhead of program development. The second supports an extended HLA software development training exercise in which groups of students work independently on components of a complete HLA simulation. An exercise of this scope becomes problematic when incomplete and possibly incorrect components are tested against each other. The tool described offers an adaptable test harness of correct components, in various states of completion, against which students can initially test their work.

1. INTRODUCTION

HLA is currently of great interest to some organisations because of its potential to greatly increase the reuse and interoperability of simulation components. As a consequence, a corresponding need exists for people to be educated in this field. However, Morse [2] notes that the number of simulation courses being offered by universities is remaining static while the simulation community is growing.

In order to address this issue a short course, presented over a two week period, was developed to teach HLA. This course was developed by members of the Australian Defence Science and Technology Organisation (DSTO). It was presented at the University of Ballarat (UOB) to a class consisting of professional programmers, wishing to gain experience in HLA, and IT Honours students who took the course as part of their degree.

The initial delivery of the course raised two important questions in the area of practical exercises. These questions were:

- How can HLA be taught to non programmers? The HLA standard is written as a set of API definitions. This tends to present the architecture as a specifically programming domain. A more general, but detailed, appreciation for non programmers is harder to achieve. In this paper we present a modification of an existing tool for evaluating the HLA.
- How can a class of students, collectively developing a distributed application, be taught? If emerging components of a distributed application are routinely tested against one another, the causes of any failure can be difficult to isolate to a single component. We present an adaptable test harness of proven components against which students can safely test their own components.

We discuss the way in which these tools can be used in the context of the strategic area of teaching distributed simulation.

2. THE HLA SHORT COURSE

The HLA Short Course was developed by John Wharington and Tony Travers from the DSTO and was first presented by them at UOB in February 2001. Since then it has been presented by the authors in February 2002 and July 2002.

The short course is presented full time over a two week period. The first week takes a lecture format, presenting the motivation for the HLA along with a detailed description of the services provided. Laboratory time is used to reinforce the lectures but actual programming tasks are avoided.

The second week is conducted entirely in the laboratory. During this week the students are required to develop a federation. This federation, called the Air Transport Operations (ATO) federation, simulates aeroplanes in various situations as they prepare for, undertake and conclude flights between airports. The ATO federation will be discussed in greater detail in a later section.

The purpose behind such significantly different weekly formats was to provide some flexibility for the

delegates. Non-programmers could usefully attend the first week only whereas developers might attend both weeks.

3. SCRIPTED TEST FEDERATE

In the first week, the lectures focus on describing each of the six sets of services defined in the HLA. Laboratory sessions were designed to expose the delegates to the actual implementation of these services by the use of a Test Federate which is designed by Kuhl et al [1]. The Test Federate is a Java application that provides menu-based access to all sections of the HLA API. Menus correspond to the six major services (Federation Management, Declaration Management etc) provided by HLA. Menu items refer to the calls within a particular service. For example, the menu item "5.2 Publish Object Class" invokes the service request of that number within the published HLA Standard.

Students use the Test Federate to exercise the particular service that is being studied, selecting menu entries and entering data into dialog boxes to cause specific HLA calls to be made. At the outset, studying Federation Management, this is straightforward, involving one or two steps but as studies progress these initial few steps must be repeated constantly in every laboratory session. The pedagogical focus on, for example, Time Management, is weakened by the need to "set the scene" through routine sets of commands in the other APIs on which the Time Management scenario depends.

The Scripting version of this Test Federate incorporates a Replay menu (see Figure 1) that enables students to manage text files that describe sequences of API calls. The Test Federate can also be configured to write to a file in which case the identity of the call (for example, "Publish Object Class") and any parameters are recorded. Alternatively the Test Federate may be directed to read back instructions from a file – in which case a previous scenario can be automatically reinstated. The text files may be edited to make alterations if required.

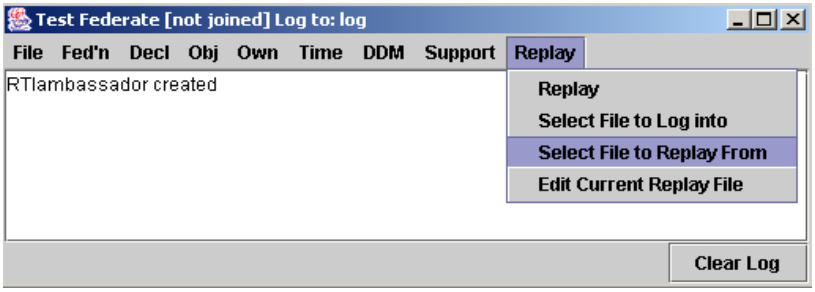

Figure 1: Test Federate "Replay" menu

As a scenario is developed the text representation of the commands that were used is written by the application to the selected text file.

Figure **2 s**hows an example of the text generated by a simple scenario with entries retrospectively numbered. The comments have been added subsequently – at this stage comments are not supported by the application. The following points should be noted:

- Handles for HLA entities, such as object classes or instances are not guaranteed to be the same across federation executions. Wherever possible names should be used as with step 6 where the object instance is given a name "myPlane"
- In practice, with the small federations used in student exercises, the previous warning about handles can often be disregarded. This is fortunate because entities such as object attributes are always un-named in HLA
- A large part of the manual use of the Test Federate consists of querying the RTI to establish current handle values as in step 3 & 4. Currently the scripting federate cannot make use of the values returned by these calls. These calls therefore play no part in the replay of this script. Extending the application to solve this problem is possible but would dramatically increase the complexity of the implementation.

Figure 3 shows the result of replaying this script into a brand new invocation of the Test Federate and the associated Pitch RTI. As can be seen the scenario has been reproduced as would an arbitrarily complex example that employed other services of the HLA – subject only to the limitations regarding the persistence of handle values.

The effect of these enhancements to the Test Federate is to provide a scripted HLA simulation environment in which non-programmers can configure very basic, but repeatable, simulations scenarios that expose all aspects of HLA API.

In the context of the broad picture of HLA pedagogy it therefore becomes possible to devise moderately extensive exercises, employing a broad cross section of HLA services, yet not requiring any programming expertise.

```
1.  createFederationExecution airport file:///g:/hla/ato.fed
2.  joinFederationExecution airport demo
3.  getObjectClassHandle ObjectRoot.position.aircraft
4.  getAttributeHandle 4 aircraft_designator
5.  publishObjectClass 4 107
6.  registerObjectInstance 4 myPlane
7.  updateAttributeValuesRO myPlane 107 JumboJet
```

Figure 2: A simple HLA scenario

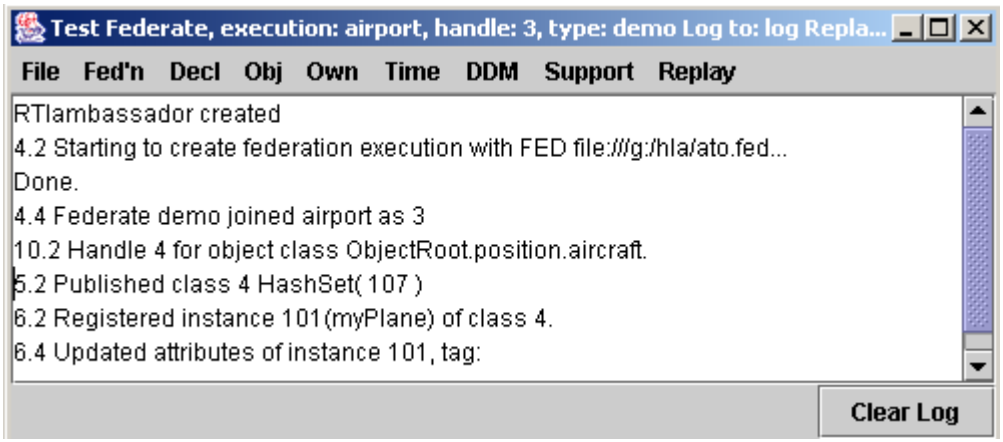

Figure 3: Result of replaying the script

4. THE TEST HARNESS

The second week of the course is spent entirely within the laboratories. The delegates are required to collectively develop an actual, non-trivial HLA simulation or federation. To complete this task the delegates are split into groups, each of which develops one component (federate) of the entire simulation. As their work progresses, they need to test the efficacy of their code against the other simulation components. In the original delivery, it was this step that proved vexatious. Evaluating untested student code against another instance of untested student code left delegates in a troubleshooting situation that simply contained too many possible causes of errors. The solution to this problem is to provide a "known to be correct" version of all federates against which student code can be tested. This section describes the way in which this test harness is configured.

4.1. THE ATO FEDERATION

The Air Transport Operations (ATO) federation was developed by John Wharington [3] for the short course. This federation models the states of one or more aeroplanes at various stages of their routine. Each aeroplane can be in one of twelve states. Some of the states that are modeled include, TAKING_OFF, LANDING, REPAIR and MAINTENANCE. State transitions occur over a period of time and are controlled by five different federates. These federates are:

- Aircraft Manager (ACM)
- Fleet Manager (FM)
- Air Traffic Control (ATC)
- Audit and Safety
- Chaos

Of these the first three are vital to the running of the federation and the fourth and fifth are optional. As the simulation proceeds, the state of each aeroplane modeled is periodically output as text to the screen. The federation has provision to handle up to one thousand aircraft in one simulation but generally it was tested with only three or four.

4.2. DEVELOPING THE FEDERATION

At the beginning of week 2, the delegates were broken up into groups. Each group was assigned a particular federate to develop. They were provided with legacy simulation code and their task was to develop an HLA compliant federate.

The original conception of the project component of the short course was to replicate, as far as possible, actual federation development conditions. As such the class was required to do the following things:

- Read and comprehend the provided specifications for the ATO federation.
- Develop an appropriate object model, in conjunction with the other groups, based on the provided

specifications. In HLA parlance, this is called the Federation Object Model or FOM.

- Using the developed FOM to modify the provided legacy code to develop an HLA compliant federate.
- To achieve milestones which were set for various times during the week. At each milestone the federates were expected to be run together as one federation execution. The groups would test and debug their federates, often in conjunction with other groups.

As has been stated this approach suffers from a considerable weakness, namely the execution of several untested, interacting components and the expectation that the source of any errors would become apparent to the students.

Another problem that we encountered was that groups were not necessarily reaching milestones at the same time. There were two reasons for this:

- The groups had differing abilities
- The federates, themselves, were of differing degrees of difficulty

The consequence of this was that the quicker finishing groups had to wait until other groups were ready before they could test and verify their programs.

To overcome these problems a test harness was developed against which groups could test their federates independently of the federates developed by other groups. The test harness is described below.

The test harness is designed to be used in the intermediate stages of federate development, to assist in the verification of student code. It is not intended to preclude the ultimate goal of a completely student written federation.

4.3. THE TEST HARNESS

The test harness provides five executable versions of correct federates against which the groups can test their own code.

The test harness is implemented as a series of Perl scripts which can be invoked through a Telnet session to the test harness server. Through Telnet a particular group of students may, through appropriate parameters, indicate what portion of the reference federation they wish to have provided by the server. In their turn, they will execute the remaining federates that are required to make up a complete simulation. In the simplest case they would be testing a single federate that they provided against four reference implementations.

At any given time the test harness server could be running several completely separate simulations corresponding to the current work needs of the groups in the class. This is a significant processing load and appropriately powerful server hardware must be selected.

The test harness had the following advantages:

- Groups which finish their work for a particular milestone can test and verify their code without the need to wait for other groups to complete their milestones.
- Groups can test their federates, confident that any errors found would be a result of errors in their own coding, rather than in another federate's code.
- The course itself can proceed even without groups to code all three of the vital federates. This provides more flexibility in presentation. If it was so desired we could have three or four groups all working on the same federate. We could also have more room to maneuver in the allocation of federates to groups.

The test harness, as it is currently implemented, also has a number of limitations.

- In order to enable the test harness to be used the groups had to be constrained to using a FOM that was provided for them rather than a FOM that they developed themselves. This significantly changes the pedagogy of the Short Course. A constructivist approach to teaching FOM would suggest that it was important that the students experience the use of the FOM that they had developed. The development of a test harness that was capable of adapting to a class developed FOM would be a significant undertaking.
- When fulfilling some of the earlier milestones, it is arguable that the testing of student code against complete reference federates is incorrect. The test harness could be extended to included correct reference federates in various levels of completion and a means to specify which particular milestone is being evaluated.
- As mentioned above, the test harness can be computationally expensive requiring the execution of the same federates in several federations. Ways of reducing computational expense include:
  - Enabling the Test Harness with as few federates as necessary. In other words, do not include unnecessary federates in a federation execution.
  - Provide incomplete federates as described above.

5. CONCLUSION

The growth of the HLA as a standard for developing distributed simulations has led to a corresponding need to produce people skilled in this area. However, Morse [2] has stated that the number of courses being offered is not growing at the same rate as the Simulation industry itself. An HLA short course, currently being run as a joint exercise between the DSTO and UOB, is designed to respond to this need.

Against this background, it is necessary to consider from a pedagogical point of view the types of learning experience that would be appropriate for course participants. The prevailing constructivist paradigm in education clearly points to practical exercises being a vital ingredient in the acquisition of complex ideas. With specific reference to distributed simulation, the provision of such exercises exposes challenges that we have sought to address in this paper.

The scripted Test federate opens significant aspects of the HLA to non trivial manipulation by non programmers.

The test harness focuses very specifically on the needs of groups that are collaborating in learning to develop components of distributed application.

We believe that these tools will make a significant contribution to the pedagogy of HLA in particular and distributed simulation in general.

Our web site describing the HLA short course is http://www.ballarat.edu.au/itms/hla.

REFERENCES

1. Kuhl, F., Weatherly, R. & Dahmann. J. (2000). "Creating Computer Simulation Systems: An Introduction to the High Level Architecture", *Prentice-Hall PTR*, Upper Saddle River, NJ 07458.

2. Morse, K. (2000) "Taking HLA Education to the Web", *Proceedings of the 2000 Winter Simulation Conference*, J. A. Joines, R.R. Barton, K Kang, P.A Fishwick (eds), pp 1619-1622.

3. Wharington, J. (2001) "High Level Architecture Tutorial Project: Air Transport Operations Federation", Short Course material.